Backstage

Keeping Backstage plugins current

with official codemods

Automate known edits. Keep the judgment.

Paul Schultz · Red Hat
Alex Bit · Codemod



Migrations are a people problem.

Confidence

What will change?

→

Trust

Can we review it?

→

Coordination

Who owns the rollout?

A shared rule helps. People still own the rollout.

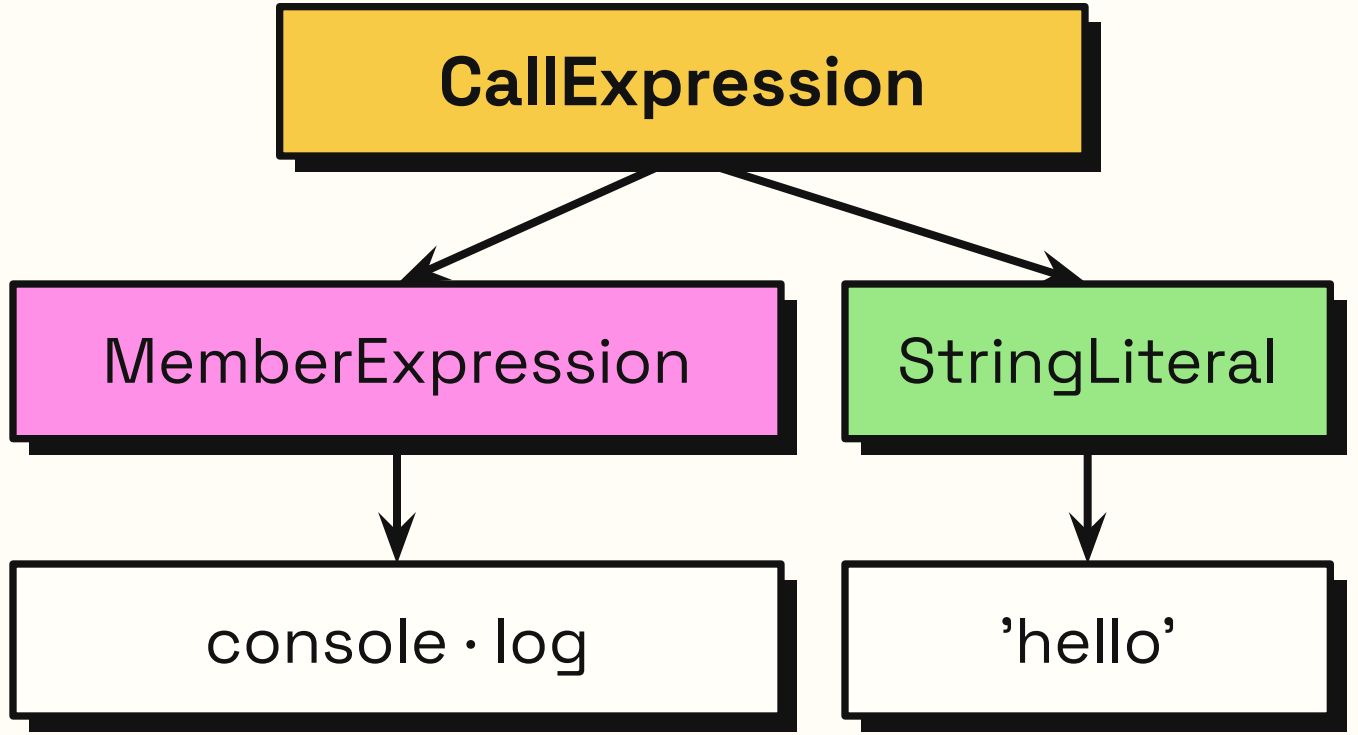
Capture the change once.



Keep expertise in code—not in every maintainer's memory.

Match structure, not strings.

```
console.log('hello');
```



An import, a call, and a string are different nodes.

Change the call.
Leave the message.

- Before

```
- console.log(
    'console.log'
  );
```

+ After

```
+ logger.info(
    'console.log'
  );
```

The matched call changes. The string stays intact.

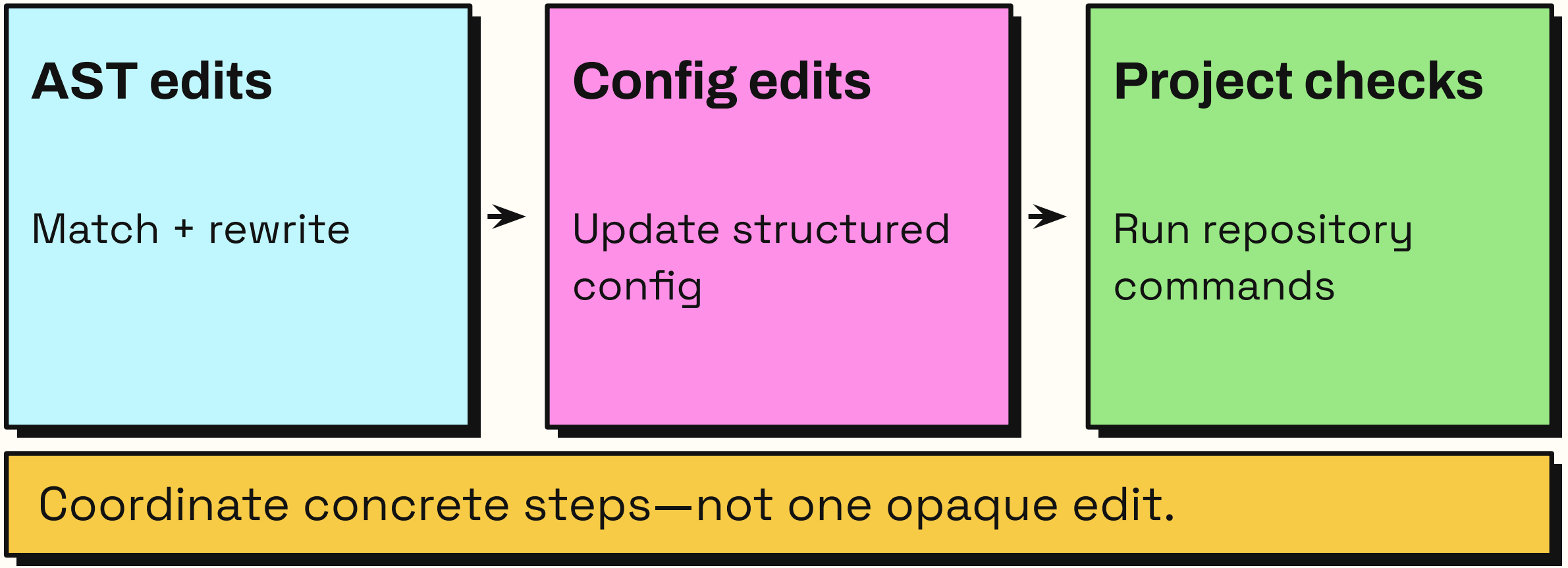
Repeatable.
Not infallible.

AI can help author a rule.

Deterministic execution applies known edits.

Neither replaces review.

One migration.
Different step types.



Author once.
Share the next rule.



Search existing rules before writing one.

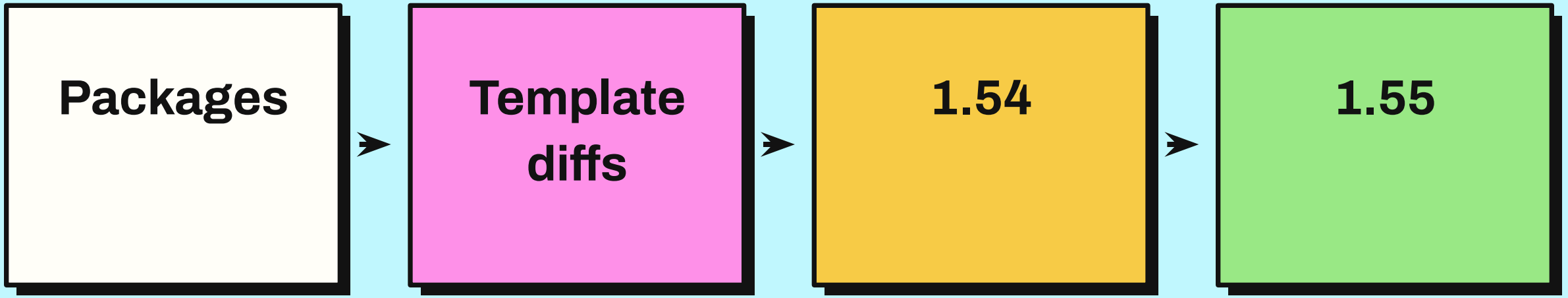
Backstage moves.
Plugins must keep up.

You opened a feature PR.

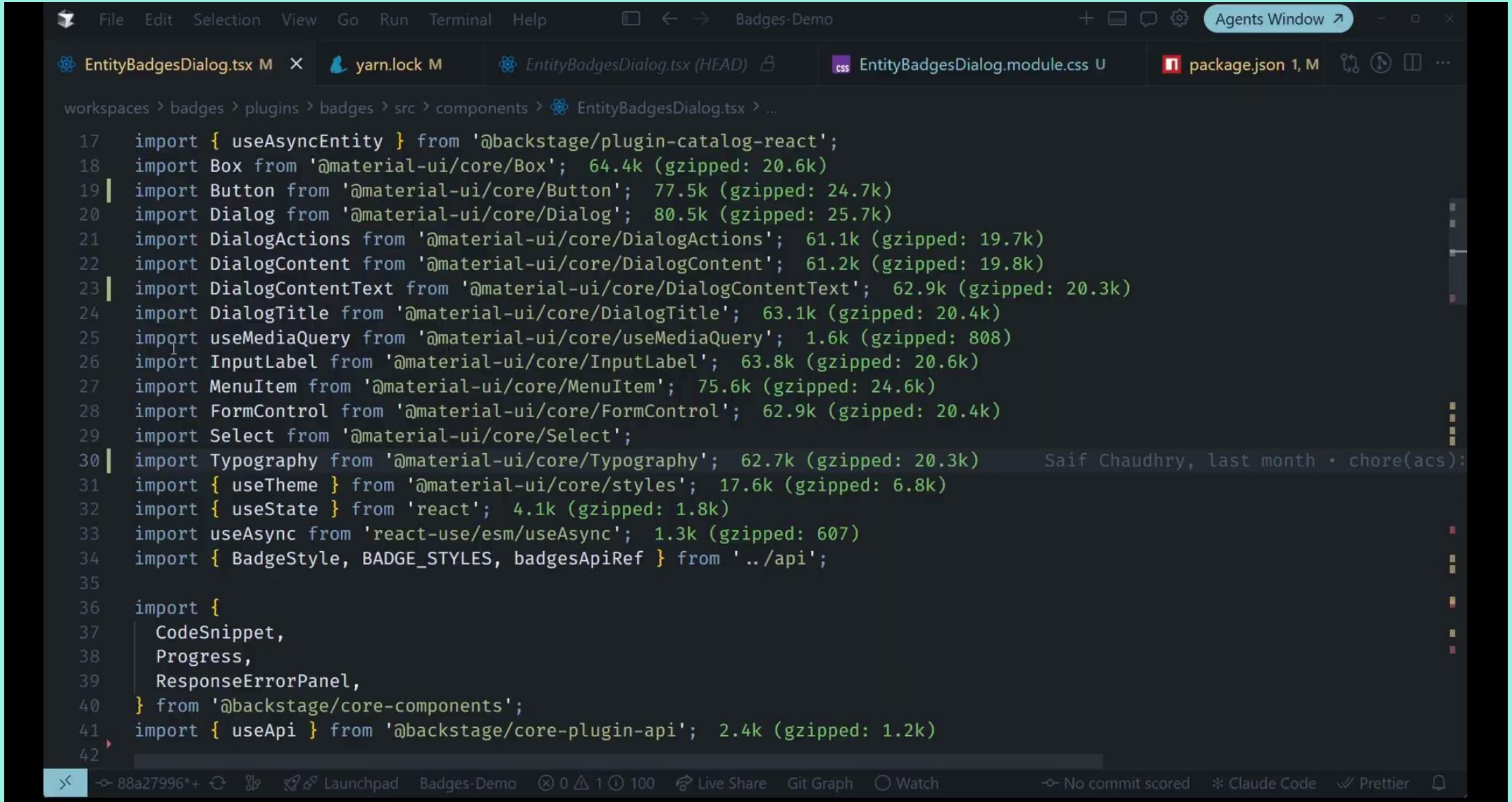
First: update the host and affected APIs.

Keep the judgment. Automate the repetition.

One plugin.
1.53 → 1.55.



Prepare the host. Then run source recipes in order.



Badges plugin · MUI → Backstage UI

Inspect the import and call.

- Input

```
 import {
-  isConnectionTypeKey
 } from
  '@backstage/connections';

 const isCatalog =
-  isConnectionTypeKey(
    'catalog'
  );
```

+ Expected

```
 import {
+  isConnectionType
 } from
  '@backstage/connections';

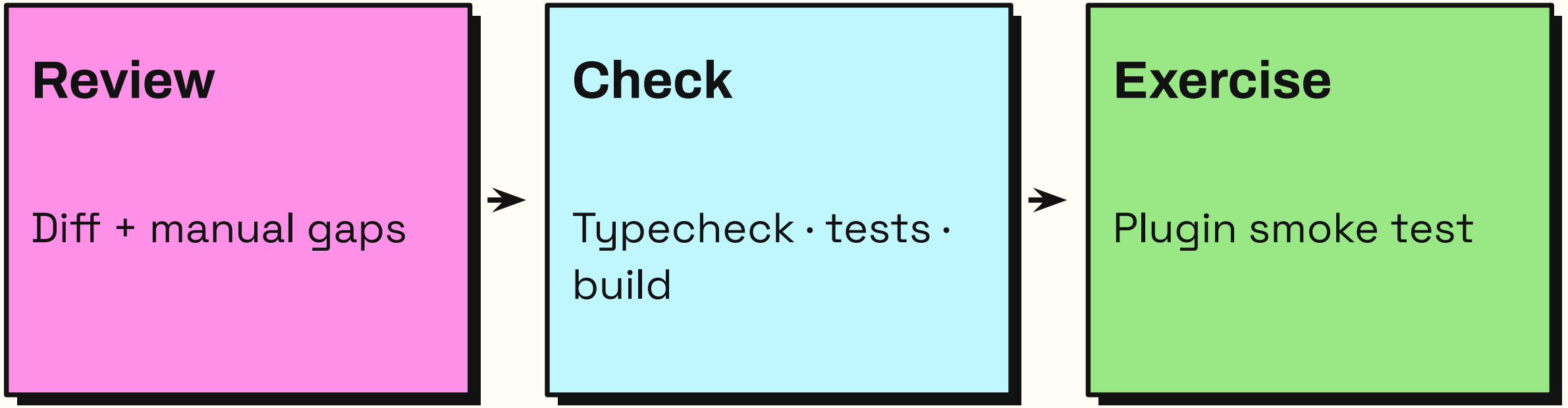
 const isCatalog =
+  isConnectionType(
    'catalog'
  );
```

Repository fixture · input → expected output

No marker ≠ no manual work.

In the code	Outside the recipe
`TODO(backstage-codemod)`	**Read the manual-only list.**
Resolve every marker.	Install · register · configure.

A changed diff is not done.

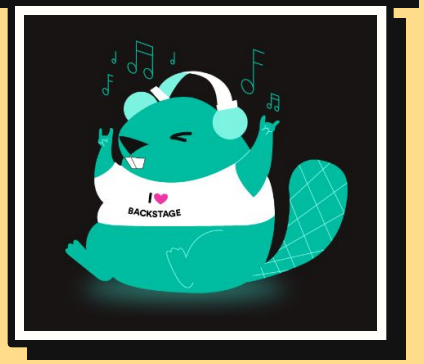


Exit code ≠ merge criterion.

Backstage

Fix one plugin.
Leave a rule.

github.com/backstage/codemods



Start with one covered upgrade.

Paul Schultz · Alex Bit